N73-19207

THE BELLFLOW SYSTEM

Stephen Pardee
Bell Telephone Laboratories

STANDARDS

One of the primary reasons for the development of BELLFLOW was the need to meet certain Bell System standards of documentation. The Bell System has been documenting designs for a long time and, in addition to trying to observe outside standards, has additional internal standards that must be observed. In the area of flowcharting, most of these additional standards relate to quality and not to symbol or line character size, legibility, titling information, change information, information placement, and format. BELLFLOW had to be able to meet these quality standards. The symbols incorporated in BELLFLOW are essentially the standard flowcharting symbols. (See fig. 1 for a list of the BELLFLOW symbols.)

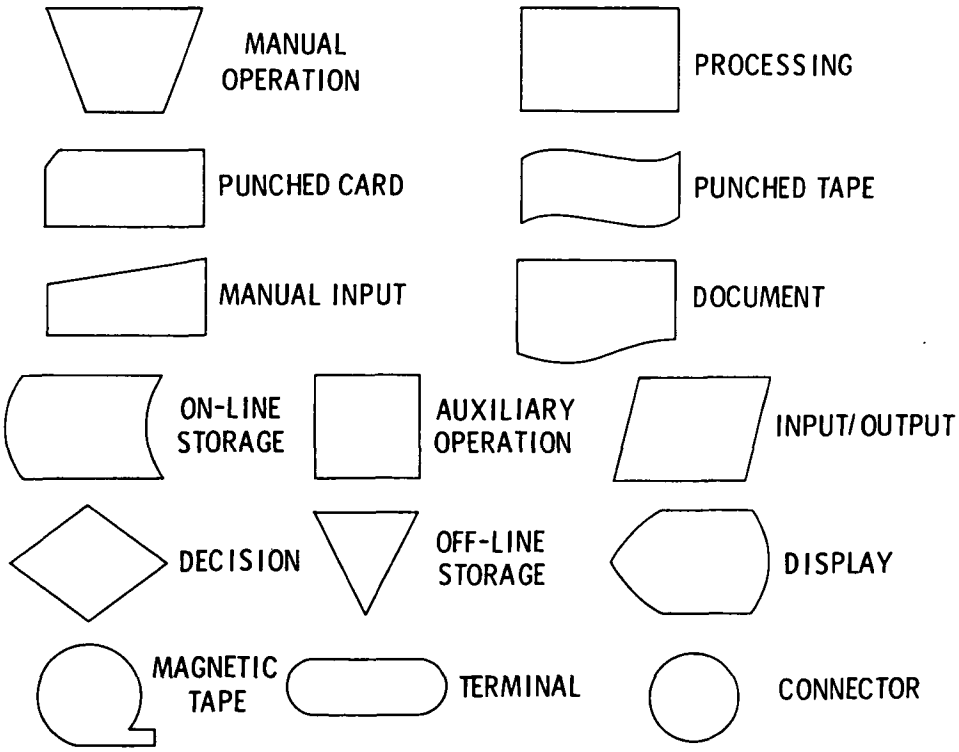


Figure 1.–Standard BELLFLOW symbols.

ADDITION OF NEW LANGUAGES

Another important reason for the development of BELLFLOW was the need to provide a flowcharting system structure that would allow the addition of new or unique special languages. There are many people at Bell Laboratories developing one-time languages or special languages that are unique to a very small area of design. These languages should be able to be conveniently added to the BELLFLOW system. By making BELLFLOW a model or table-driven system, a language has been developed that can be used for defining new programming languages in terms of BELLFLOW. As an example, consider the standard IF statement:

```
IF A = B THEN C = D
```

In BELLFLOW language, this would be

```
.B.IF.B..T..B.THEN.B..S.  /S(B*)
```

The following shows the entire set of model language statements required to define a programming language that is very close to PL/I structurally but not quite as complicated:

```
    .STATEMENT(";").
    .LABEL(":").
    .COMMENT("/*").
    .COMMENT(,"*/").
    (.B.IF.B..T..B.THEN.B..S.)    /S(*B)
    (B.ELSE.B..S)                 /S(*L)
    (.B.GObTO.B..J.)              /S(*Z)
    (.B.DO.E.=.E.)                /S(*D)
    (.B.DO.B.)                    /S(*G)
    (.B.END.B.)                   /F(F1)
    (.B.END.B..J.)                /(*T)
F1  (.B.CALL.B)                   /S(*C)
    (.B.RETURN.B.)                /S(*Y)
    (.B.EXIT.B.)                  /S(*S)F(*X)
```

This set of statements is not supposed to provide an understanding of the details of the language itself but to show how straightforward it is to define a new programming language for the BELLFLOW system. For example, a new language called CENTRAN was recently added to BELLFLOW. It took about a week to develop the models, and, within another week or two, the system was essentially debugged and ready for use by programmers writing in the CENTRAN language.

USE OF GRAFPAC

Bell Laboratories uses a software package called GRAFPAC, which interfaces with all of our graphical devices. At the present time, the GRAFPAC system supports such devices as the SD 4060 and FR 80 microfilm plotters; the CALCOMP 718 and 728 and EAI 3500 line plotters; the Graphic 101 CRT terminal: STARE, an on-line hard-copy graphic unit that

possesses quick turnaround features; and several others. The flowcharting system is interfaced through GRAFPAC so that, as new graphical devices are brought into Bell Laboratories and added to the GRAFPAC system, they are automatically available to the BELLFLOW system.

MULTIPLE-LEVEL FLOWCHARTS

Multiple-level flowcharts (very-high-system-level and very-low-detail-level) should also be imbeddable in the same source program. In addition, it is often desirable to imbed a separate flowchart for each procedure of a program even though all the procedures are compiled as a unit.

MODES OF OPERATIONS

There are three modes of operation: source mode, comment mode, and mixed mode. In the source mode, all of the flowcharting information is derived directly from the source code. This is similar to the way many other flowcharting systems operate. There is no reliance on comments or other information in the derivation of the entire flowchart.

In the comment mode, BELLFLOW ignores the source code completely and derives the entire flowchart purely from comments imbedded in the program. This has the advantage of being able to imbed comments in any program, whether or not the source language is supported by BELLFLOW. In the mixed mode, the source and comment mode are combined. The text that is to appear within a symbol and the definition of the symbol to be used are derived from a comment imbedded in the source code, but BELLFLOW uses the source code itself to determine the connections among the symbols and the placement and layout of the flowchart.

SELF-DOCUMENTING SOURCE DECK

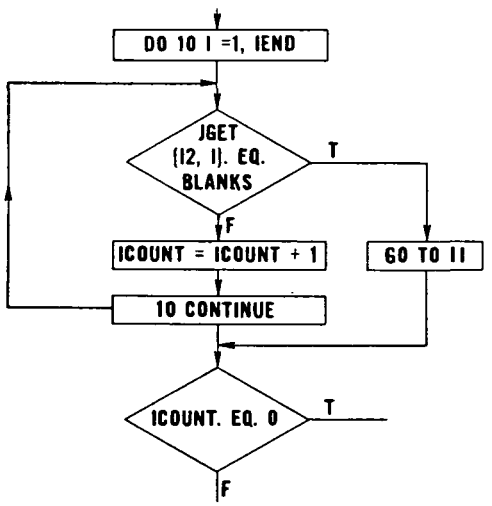
Because the mixed mode is unique to BELLFLOW, it may be of interest to explore the particular advantages of this mode. The goal was to try to provide a self-documenting program source deck. To achieve self-documentation, the flowchart should be imbedded in the source deck, and the source deck must be well commented. The question of what constitutes a "well-commented" program is too difficult to bring up here. In any event, most programmers use an intuitive definition of the term in deciding whether a program is well commented. A brief program description should also be imbedded at the beginning of the source deck. Given these three features, a subroutine or a program would contain most of the information needed for program documentation.

If the program flowchart is based on the source code alone (using the source mode), a very good two-dimensional representation and a poor functional description of the program are obtained. Many people feel that FORTRAN and other languages, such as PL/I, can be documented very nicely by placing the source code within the symbols, but this approach does not seem very fruitful.

If the flowchart is based only on comments that are imbedded in the program (using the comment mode), the result may be an excellent functional description of the program

```
      DO 10 I = 1, IEND
      IF (JGET (I2, I) .EQ. BLANKS) GO TO 11
      ICOUNT = ICOUNT + 1
10    CONTINUE
11    IF (ICOUNT .EQ. 0) ICOUNT = 1
```

(a)



(b)

Figure 2.—(a) FORTRAN source mode. (b) Corresponding flowchart.

in understandable language, but no real tie-in between those comments and the program itself. It is not certain that the comments reflect the flow and interconnectivity of the program. Furthermore, the programmers usually have to supply a lot of redundant information in the comments to express the interconnectivity among the symbols.

The mixed mode permits one to put text inside symbols in a way that is meaningful to a person trying to understand a program, and yet, the fact that the program flow (the interconnectivity among symbols) is derived from the source code itself is still retained. No other comments other than the BELLFLOW comments are required to produce a well-commented program. If a preface is added at the beginning of the program (a manual operation), a self-documenting source program deck is obtained.

EXAMPLES OF BELLFLOW

Figure 2(a) is an example of the source mode in FORTRAN. Figure 2(b) shows the corresponding flowchart that would be generated. Quite often, the source mode is used when improvements have to be made in an old FORTRAN program. Because it is not always possible to locate the original program writer, it is very convenient to be able to obtain a two-dimensional flowchart listing in the source mode; this makes the task much easier than it would be if a linear listing of the program had to be used.

The comment mode format is

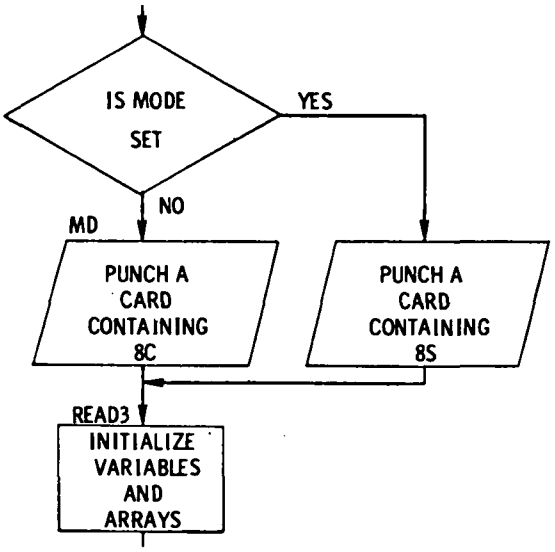
```
*F(LEVEL)(LABEL)  TEXT  /SYMBOL/OUTPUT
```

The comment begins with an indicator that is the normal comment indicator of the programming language. For an assembly language, the indicator might be an asterisk in the first column. The letter F indicates that a flowchart comment is being made. The next field is a level indicator that allows one to imbed more than one set of flowchart comments; it is a two-character identifier that denotes a particular flowchart. If there is only one flowchart imbedded in the program, this field may be left blank. The label field is usually left blank unless it is needed to specify interconnectivity.

The main body in this statement is the comment text field, which conveys the information that is to appear in the symbol. The final two fields at the end of the statement are the symbol field and the output field. The output field contains information that is to be placed

```
*F          IS MODE SET                          /D/NO(MD)YES
*F          PUNCH A CARD CONTAINING 8S           /TO/(READ3)
*F  MD      PUNCH A CARD CONTAINING 8C           /TO
*F  READ3   INITIALIZE VARIABLES AND ARRAYS
```

(a)



(b)

Figure 3.—(a) Imbedded comments. (b) Typical comment mode flowchart.

on output branches emanating from symbols, and, in some cases, it specifies the interconnectivity between one symbol and another.

An example of the comment mode format is

```
* F IS COUNT=MAX+TOL  /D/YES,NO
```

Here, the level is blank, and there is no label. The letter D in the symbol field indicates a decision symbol. One branch will have the label YES, and the other branch will have the label NO. Figure 3 shows a typical portion of a comment mode flowchart.

The comment mode is extremely useful in preliminary design for the generation of system flowcharts when no code exists or as part of the documentation of an early design. A complicated flowchart can be quickly encoded, run off, modified, edited over a period of meetings, and kept up to date by using the comment mode without ever having any source code. The comment mode is also being used to generate nonprogramming documentation such as flowchart-like and program evaluation and review technique drawings, and input/output relation tables.

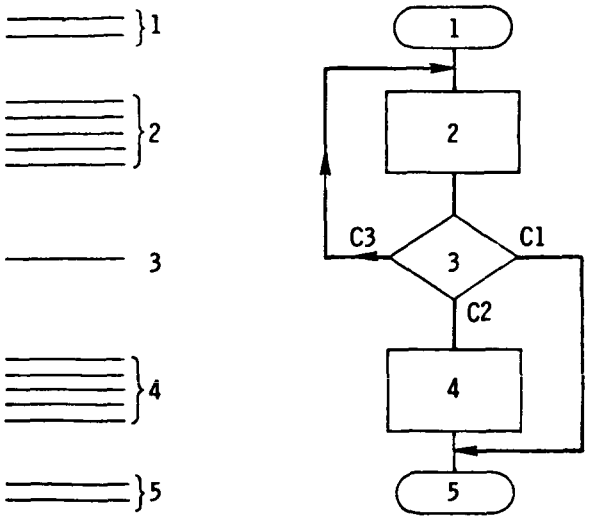


Figure 4.–Correspondence of program to flowchart.

Figure 4 is a flowchart consisting of five symbols. The lines on the left are used to indicate statements of source code and are grouped to indicate which statements correspond to the five symbols in the flowchart. In the mixed mode, a BELLFLOW comment is inserted at the beginning of each one of these functional groupings of source code.

Figure 5(a) is an example of the mixed mode. Note that the comments are very similar to the ones that might be placed in a program even if BELLFLOW were not being used. The second BELLFLOW comment, "ARE SIGNS EQUAL," is the text for a decision symbol with two branches labeled YES and NO. There are two ways to exit from this functional block. The first one is encountered at the skip instruction (SPA), and the second one is encountered at the jump (JMP), so the YES text is associated with the first way to exit and the NO text is associated with the second way to exit. Figure 5(b) shows the corresponding flowchart. It is important to repeat at this point that the BELLFLOW comments are really all the comments that are needed to provide a well-commented source deck.

BELLFLOW FEATURES

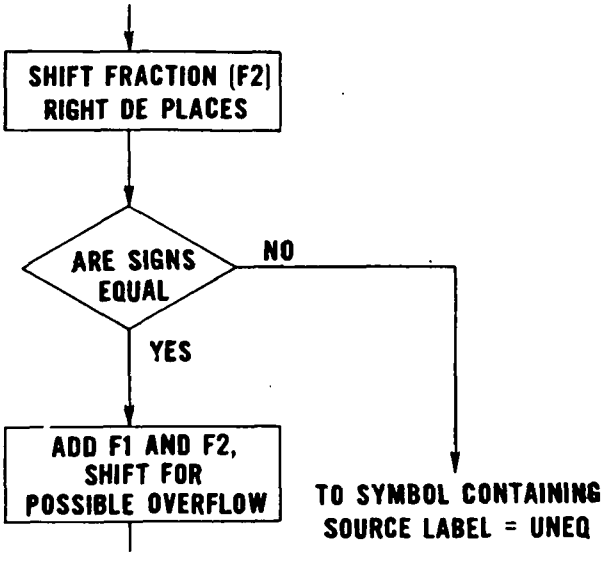
In addition to automatic placement, automatic line routing, paging, and generation of on- and off-sheet connectors, all of which are standard regardless of the mode employed, BELLFLOW possesses some additional features. One can request left-to-right rather than top-to-bottom flow in the flowchart. Normally, BELLFLOW will automatically format the text that goes in a symbol, but the programmer can do this himself if he considers it desirable.

BELLFLOW will adhere to standard aspect ratios in choosing symbol sizes but also has the capability of using nonstandard symbol sizes. In the latter case, processing symbols (rectangles, for example), are reduced to fit exactly around the text provided. It is interesting to note that, in a brief study of this capability, a gain of almost 50 percent in the number of symbols per vertical row was achieved with the use of the nonstandard symbol sizes. Increasing the number of symbols per sheet reduces the number of off-sheet connectors and, therefore, simplifies the flow and makes it more readable.

BELLFLOW permits multiple keypunch codes to be used, and an option is provided that permits flowcharts to be made without any crossovers. The normal mode will make crossovers unless too many lines are crossed, in which case an on-sheet connector is generated.

```
        DAC     F2
*
*F              SHIFT FRACTION (F2) RIGHT DE PLACES
*
SHIFT   LAC     DE
        TAD     SHIFT1
        DAC     **2
        LAC     F2
        DSS     1
        DAC     F2
*
*F              ARE SIGNS EQUAL                         /D/YES,NO
*
        LAC     F1
        XOR     F2
        SPA
        JMP     UNEQ
*
*F              ADD F1 AND F2,SHIFT FOR POSSIBLE OVERFLOW
*
        CLL
         .
         .
```

(a)



(b)

Figure 5.–(a) Comments. (b) Flowchart for BELLFLOW mixed mode example.

SAMPLE OUTPUTS

Figure 6 is an example of the BELLFLOW output produced by the FR 80 microfilm plotter. Note the placement, line routing, and general quality features of the flowchart. The cost of such a flowchart sheet, including the analysis and drawing, is about $2 to $3 when the IBM 360/85 is used.

Figure 7 is another example of the BELLFLOW output; this one was made by a CALCOMP line plotter. It is larger than the sheet shown in Figure 6 and has about 50 symbols. Again, the important things to note are the placement, line routing, and general quality features of the flowchart. Figure 8 shows the same flowchart that Figure 7 does, but it was made with left-to-right rather than vertical flow.

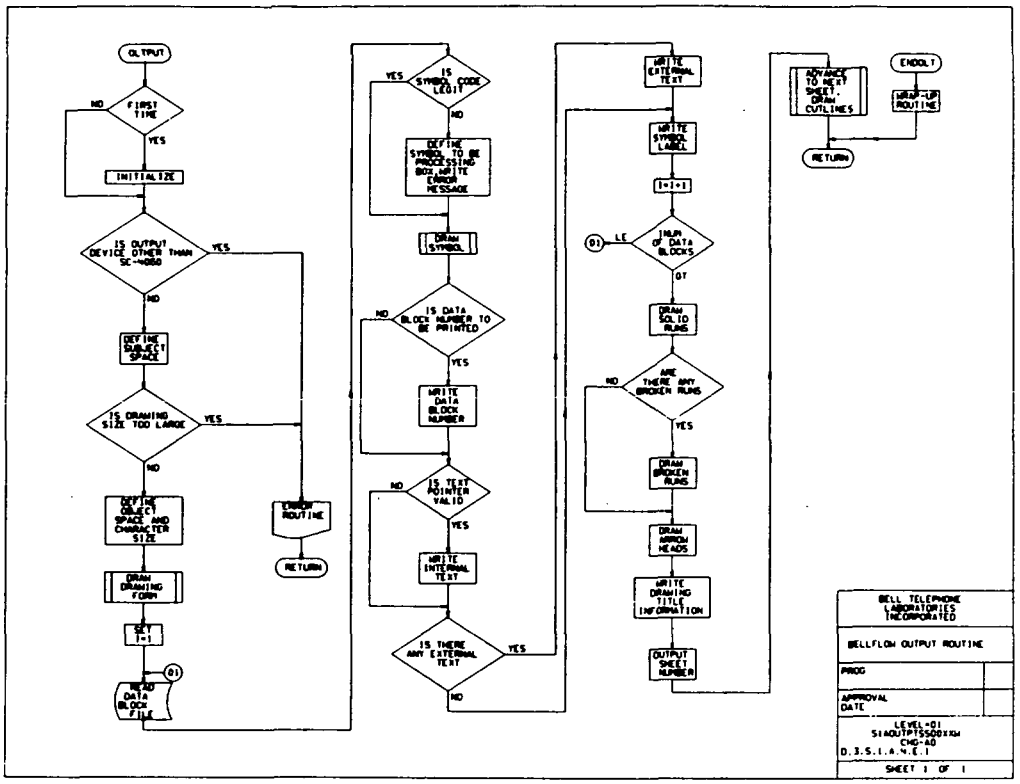


Figure 6.–Sample FR 80 microfilm output.

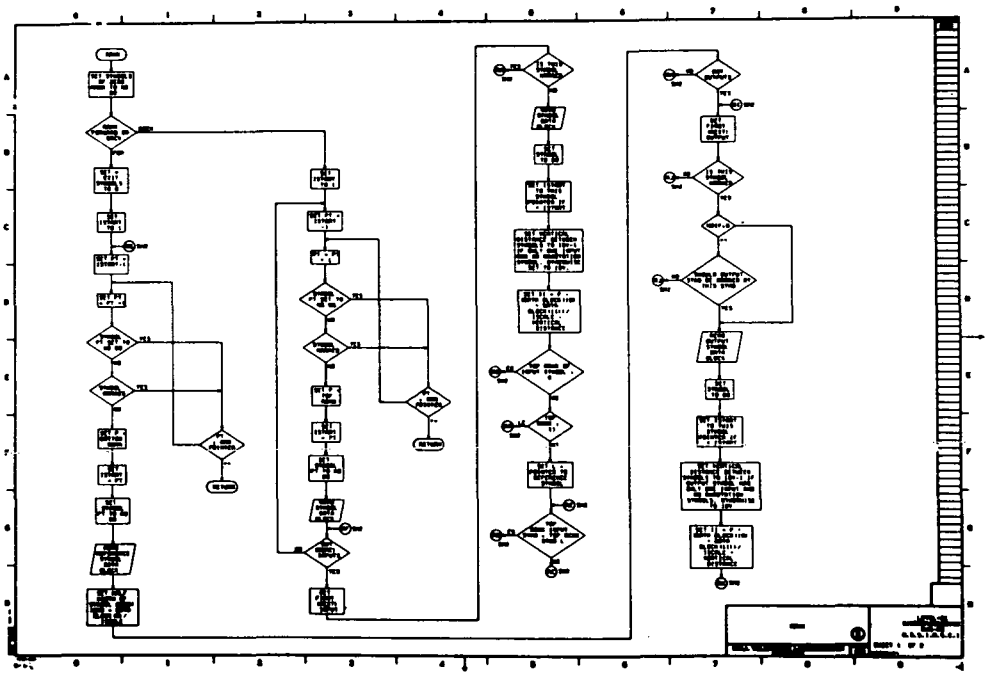


Figure 7.–Sample CALCOMP output.

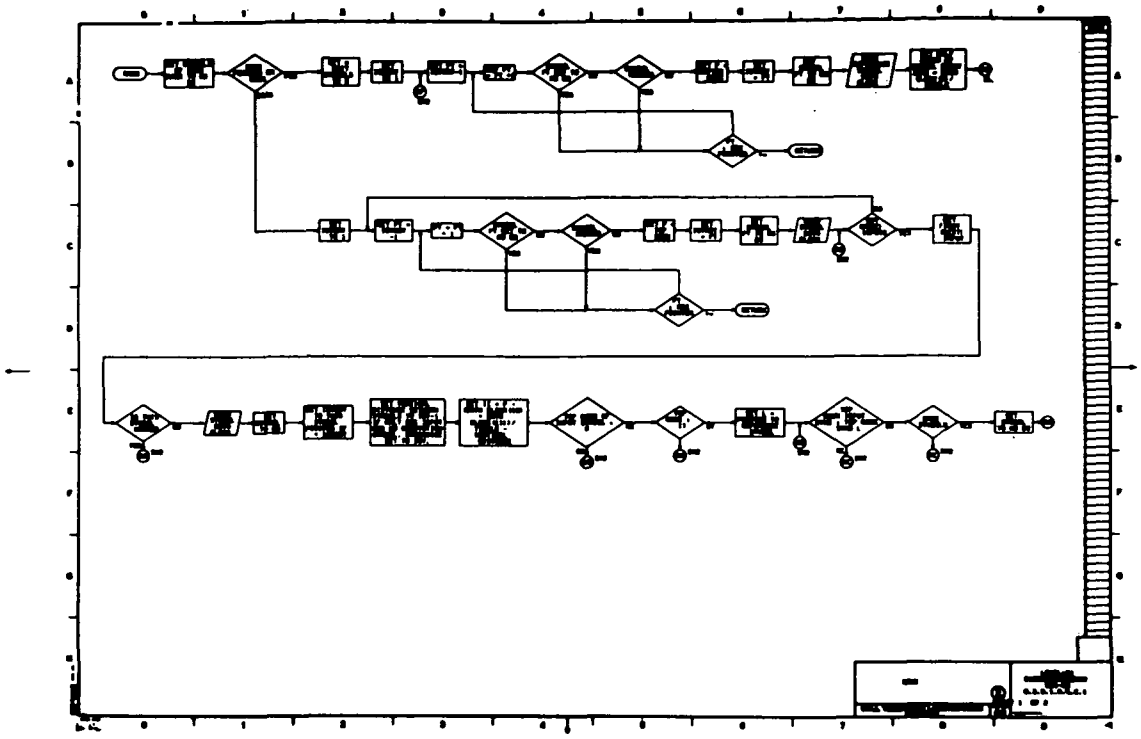


Figure 8.–Sample CALCOMP output, left-to-right flow.

CONCLUSION

The BELLFLOW system has been working for about a year and a half. One project manager, who heads a rather sizable software development project, stated that with BELLFLOW he felt that he had an adequate means for insuring that he had both a well-commented and properly flowcharted program. He insists that programmers must deliver a flowchart, in the mixed mode, to him before he will accept their program and consider it complete. By following this procedure, he reasons that, if a good flowchart can be produced, then the comments that appear in the deck are also understandable, meaningful, and useful. So he accomplishes the dual goal of having a program well-commented and properly flowcharted by using BELLFLOW as an administrative tool.

DISCUSSION

MEMBER OF THE AUDIENCE: In the event that the comments and the actual operation statements do not match, is there an editing function in the BELLFLOW system that would point this out?

PARDEE: There is no specific function that points this out.

MEMBER OF THE AUDIENCE: I would like to ask the same question I asked Goetz: Do you document BELLFLOW with BELLFLOW?

PARDEE: Yes.

MEMBER OF THE AUDIENCE: If the syntax of the modal language is unambiguous in the program mode, would it be possible to program from the flowcharts using the logic to work back to a program? Also, is it possible to alter the placement of things on flowcharts?

PARDEE: Programming from the flowchart is something that we are going to study. We do not know enough yet to be able to say whether it might be helpful to input a flowchart and then have the machine automatically convert the logic into a source code. There is a manual placement option that can exert a certain amount of control in the relative placement of symbols, but it does not control specific placement. We are investigating the possibility of a manual touchup capability that would be able to make specific changes in the flowchart.

MEMBER OF THE AUDIENCE: Why was not any mention made of the IBM 360/67?

PARDEE: Our philosophy was to detach ourselves from the compiler and assembler functions to remain somewhat independent of the machines and any changes involving compilers and assemblers and to deal only basically with the language, which was fairly well defined.

MEMBER OF THE AUDIENCE: Do you have any plans for making BELLFLOW a commercial product?

PARDEE: No. There is the possibility of people having access to BELLFLOW, however.